Name	Kang Chin Shen
Lecture	CS326 Agent Based Modelling & Simulation (G2)
SMU Email ID	cskang.2020@scis.smu.edu.sg
SMU Student ID	01412921

CS426 Assignment 3

Problem 1

ASSUMPTIONS

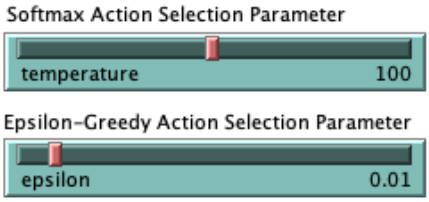
1. Definition of "Equilibrium" (consensus) in this game:
   * Based on the coordination game payoff's matrix, agents prefer to use the same color as their opponent. Hence, an equilibrium state is defined as ALL agents converge to a single action in 30,000 ticks (i.e., ONLY one color on the visualisation).
   * The agents may converge to either one of the actions (or colors) as the payoff is identical for both (A, A) and (B, B).
   * The Netlogo program checks if the convergence requirement has been met at each iteration, stop the program execution upon successful convergence and print the following message on the console:

```
### SUCCESSFULLY CONVERGED! ###
```

2. Definition of "History":
   * An agent, x always "pool" all its past plays against his neighbors into a single history (to be stored as agent-level variables).
   * This implies x is not dependent on the identities of its opponents, as it treats all opponents as the same and react based on its past experience of all plays.
3. Definition of "Random Selection":
   * At each iteration, an agent, x randomly selects one of its immediate neighbors to player with (see the visualisation below).

n1	n2	n3
n8	x	n4
n7	n6	n5

   * Every agent will have its chance to choose and play against an opponent, even it has already been chosen as other agent's opponent in the same round.
4. Two adaptation processes are set up here, i.e., Fictitious Play and Reinforcement Learning (Cumulative Propensity Matching Learning).
   * For Reinforcement Learning, there are additional parameter added to the UI for an observer to experiment with:
     - An observer can choose to use either "Epsilon Greedy" or "Softmax" as action selection approach.
     - Each action selection strategy has its corresponding parameter:



     - Note that for "Softmax", a low temperature value could potentially cause the Netlogo system to crash as to the exponent function takes in large input used in the "Softmax" function. Hence, the default value is set to 100, which has been tested to work decently (converge 50% of the time).
     - For this particular action selection approach, the model automatically stop execution after 30,000 ticks to avoid Netlogo from crashing (if you set temperature to any value below 100, Netlogo might crash before 30,000 ticks).
     - The following message will be displayed if the model executes more than 30,000 ticks.

```
**********************************************************************************
The model is stopped automatically before reaching Netlogo computation threshold.

TRY one of the following:
1. Run a few more times to see if convergence is possible.
2. Adjust the 'temperature' slider.
3. Use Epsilon-Greedy action selection instead.

**********************************************************************************
```

5. White color represents an agent selected the action "A" and blue color represents an agent selected the action "B".
6. The Netlogo model is set to have 10 x 10 agents participating in this coordination game.
7. The Netlogo model will randomly assign each patch an action (A or B) on "setup", there should be around 50% of agents choosing each action at $t = 0$.
8. The "Comment On The Effectiveness" part is based on the following configuration:
   * Fictitious Play
   * Reinforcement Learning (CPM) with Epsilon-Greedy
     - $\epsilon = 0.01$
     - Softmax is NOT chosen due to limited computation power of Netlogo. Lower τ value may lead to better performance (convergence may happen in few ticks) but it is NOT possible for Netlogo to store $e^{\theta_{s_x}^t/\tau}$ when $\theta_{s_x}^t$ grows.

Payoff Matrix

	A	B
A	1,1	-1, -1
B	-1, -1	1, 1

FICTITIOUS PLAY – Action Selection Description

n1	n2	n3
n8	x	n4
n7	n6	n5

1. Probability of an agent n plays a strategy $s_n \in S$ where $S = \{\text{A, B}\}$ at an iteration, t is computed as follows:

$$P_n^t(s_n) = \frac{k_n^t(s_n)}{\sum_{\hat{s}_n \in S} k_n^t(\hat{s}_n)} \; where$$

$$k_n^t(s_n) = k_n^{t-1}(s_n) + \begin{cases} 1 \; if \; s_n^{t-1} = s_n \\ 0 \; if \; s_n^{t-1} \neq s_n \end{cases}$$

Let the target agent, x randomly pick one of its neighbors as the opponent agent (i.e., $n1$ for this case):

* At $t = 0$, $k_x^0(A) = k_x^0(B) = k_{n1}^0(A) = k_{n1}^0(B) = 0$.
* Netlogo model will randomly assign each patch an action (A or B) on "setup". For simplicity, here we assume agent x played "A" and $n1$ played "B" at $t = 0$.
* At $t = 1$, $k_x^1(A) = 1$, $k_x^1(B) = 0, k_{n1}^1(A) = 0, k_{n1}^1(B) = 1$.

2. Agent x remembers the historical distribution of ALL its plays, and compute the best strategy when at $t = 1$ by selecting whichever action that yields higher expected utility, which is computed as below:

$$EU(s_x) \; at \; t = \sum_{s_n \in S} P_n^t(s_n) \cdot U(s_x, s_n)$$

Continue from 1., suppose the target agent x randomly pick $n1$ as the opponent agent again, then based on the historical distribution of its plays, it computes the expected utility for each action, and chooses the action that yields higher value.

* Calculate $EU(A) = P_{n1}^1(A) \cdot U(A, A) + P_{n1}^1(B) \cdot U(A, B) = 0 \cdot 1 + 1 \cdot (-1) = -1$
* Calculate $EU(B) = P_{n1}^1(A) \cdot U(B, A) + P_{n1}^1(B) \cdot U(B, B) = 0 \cdot (-1) + 1 \cdot 1 = +1$
* Hence, agent x "thinks" B is the best strategy to use against agent $n1$ at $t = 1$ as it yields higher expected utility.
* Similar approach is taken by the opponent agent, and the process repeats for all agents until consensus is reached.

3. Obliviousness :

* Agent x does NOT select its action based on the identity of its opponents. For this case, x does NOT select action "B" at $t = 1$ because of the identity of $n1$. Instead, x computes expected utility based on ALL of its previous plays and select the action that yields highest value.
* Agent x does NOT select its action based on the name of actions. Whenever there is a tie in expected utility, x randomly break it instead of always choosing a specific action.

```
(ifelse utility-A > utility-B [
  set strategy "A"
  set pcolor strategy-color-A
] utility-A < utility-B [
  set strategy "B"
  set pcolor strategy-color-B
][
  ifelse random-float 1 < 0.5 [
    set strategy "A"
    set pcolor strategy-color-A
  ] [
    set strategy "B"
    set pcolor strategy-color-B
  ]
])
```

4. Locality:

* Agent x selects action based on agent x's personal experience and NOT based on any global system properties.
* Agent x is semi-local, based on the definition in Assignment 3's definition.

REINFORCE LEARNING (CUMULATIVE PROPENSITY MATCHING LEARNING) – Action Selection Description

n1	n2	n3
n8	x	n4
n7	n6	n5

1. Cumulative Propensity Matching is defined as below:

$$\theta_{s_x}^{t+1} = \begin{cases} \theta_{s_x}^{t} + U(s_x, s_{opponent}) & if\ s_x\ is\ played\ at\ iteration\ t \\ \theta_x^t & otherwise \end{cases} \quad where\ s_x \in \{\mathrm{A}, \mathrm{B}\}$$

Let the target agent, x randomly pick one of its neighbors as the opponent agent (i.e., $n1$ for this case):
* At $t = 0$, Netlogo model initialize $\theta_A^0 = 1, \theta_B^0 = 1$ for agent x.
* Netlogo model will randomly assign each patch an action (A or B) on "setup". For simplicity, here we assume agent x played "A" and $n1$ played "B" at $t = 0$.
* At $t = 1$, $\theta_A^0 = 0, \theta_B^0 = 1$ for agent x.

2. Agent x remembers the propensity of each action of ALL its plays and compute the best strategy when $t = 1$ by assigning relatively higher probability to the action with higher propensity.
* The "relativity" of probability depends on the action selection function that an observer chooses to use:

(a) Epsilon-Greedy
- At iteration t, probability $P_{s_x}^t$ of an agent x plays a strategy $s_x \in S$ where $S = \{\mathrm{A},\ \mathrm{B}\}$, t is computed using ϵ-Greedy action selection:

$$P_{s_x}^t = \begin{cases} 1 - \epsilon & if\ s_x\ is\ the\ best\ strategy\ (highest\ propensity) \\ \epsilon & otherwise \end{cases}$$

- For this case, agent x has a probability of $1 - \epsilon$ choosing action "B" and ϵ probability of choosing "A", given that ϵ is small.

(b) Softmax
- At iteration t, probability $P_{s_x}^t$ of an agent x plays a strategy $s_x \in S$ where $S = \{\mathrm{A},\ \mathrm{B}\}$ is as follows:

$$P_{s_x}^t = \frac{e^{\theta_{s_x}^t/\tau}}{\sum_{s_y \in \mathrm{S}} e^{\theta_{s_y}^t/\tau}}$$

- For this case, agent x has a $\frac{e^1}{e^1+e^0} = 0.73$ probability of choosing action "B" and $\frac{e^0}{e^1+e^0} = 0.27$ probability of choosing "A", given that $\tau = 1$.

* Hence, agent x "thinks" B is the best strategy to use against agent $n1$ at $t = 1$ (for both Epsilon-Greedy and Softmax).
* Similar approach is taken by the opponent agent, and the process repeats for all agents until consensus is reached.

3. Obliviousness :
* Agent x does NOT select its action based on the identity of its opponents. For this case, x does NOT assign higher probability to action "B" at $t = 1$ because of identity of $n1$. Instead, x computes propensity of corresponding actions based on all previous plays and assigns a probability score to each of those actions based on the action selection approach chosen by an observer.
* Agent x does NOT select its action based on the name of actions. Whenever there is a tie in expected utility, x break it randomly break it instead of always choosing a specific action.

```
ifelse action-selection = "Epsilon Greedy" [

  ;; Epsilon Greedy
  (ifelse strategy-propensity-A > strategy-propensity-B [
    ifelse random-float 1 < 1 - epsilon [
      set pcolor strategy-color-A
      set strategy "A"
    ][
      set pcolor strategy-color-B
      set strategy "B"
    ]
  ] strategy-propensity-A < strategy-propensity-B [
    ifelse random-float 1 < 1 - epsilon [
      set pcolor strategy-color-B
      set strategy "B"
    ][
      set pcolor strategy-color-A
      set strategy "A"
    ]
  ][

    ;; Random selection to break the tie
    ifelse random-float 1 < 0.5 [
      set pcolor strategy-color-B
      set strategy "B"
    ][
      set pcolor strategy-color-A
      set strategy "A"
    ]
  ])
] [

  ;; Softmax
  let probability-strategy-A (exp (strategy-propensity-A / temperature)) / ((exp (strategy-propensity-A / temperature)) + (exp (strategy-propensity-B / temperature)))
  let probability-strategy-B (exp (strategy-propensity-B / temperature)) / ((exp (strategy-propensity-A / temperature)) + (exp (strategy-propensity-B / temperature)))

  (ifelse random-float 1 < probability-strategy-A [
    set pcolor strategy-color-A
    set strategy "A"
  ] [
    set pcolor strategy-color-B
    set strategy "B"
  ])
]
```

4. Locality:
   * Agent x selects action based on agent x's personal experience and NOT based on any global system properties.
   * Agent x local, based on the definition in Assignment 3's definition.

Comment On The Effectiveness

1. 20 rounds of experiment are conducted to compare Fictitious Play and Reinforcement Learning (CPM) with Epsilon-Greedy ($\epsilon = 0.01$). The following is the result of the experiment:

[run number]	adaptation-process	[step]	count patches with [pcolor = blue] = 100 or count patches with [pcolor = white] = 100	Average Number Of Ticks
3	fictitious play	1168	TRUE	1628
10	fictitious play	2111	TRUE	
17	fictitious play	385	TRUE	
18	fictitious play	116	TRUE	
11	fictitious play	4360	TRUE	

Fictitious Play

[run number]	action-selection	adaptation-process	[step]	count patches with [pcolor = blue] = 100 or count patches with [pcolor = white] = 100	Average Number Of Ticks
1	Epsilon Greedy	reinforcement learning	143	TRUE	9923.625
6	Epsilon Greedy	reinforcement learning	5654	TRUE	
15	Epsilon Greedy	reinforcement learning	1033	TRUE	
14	Epsilon Greedy	reinforcement learning	12294	TRUE	
13	Epsilon Greedy	reinforcement learning	12634	TRUE	
18	Epsilon Greedy	reinforcement learning	5757	TRUE	
12	Epsilon Greedy	reinforcement learning	16949	TRUE	
19	Epsilon Greedy	reinforcement learning	24925	TRUE	

Reinforcement Learning (CPM) with Epsilon-Greedy ($\epsilon = 0.01$)

2. Based on the experiment, 25% of the time Fictitious Play converge within 30,000 ticks, and 40% of the time Reinforcement Learning converges within 30,000 ticks.
   * This implies that Reinforcement Learning has higher chance of achieving convergence under the current setup.
   * This could be due to Fictitious Play has a relatively higher chance to be prone to local optimal, as there is no parameter to adjust the degree of exploration. The decision making is purely based on the utility function defined.
3. In general, although Reinforcement Learning has a higher convergence probability, it could potentially suffer from the mixed strategy's behavior. For example,
   * An agent is exploring relatively less over each iteration if the ϵ is set to a relatively small value.
   * On the other hand, setting ϵ is set to relatively high value could potentially cause an agent that has already been executing the correct "action" to make wrong decision.
   * This is clearly illustrated above as the average number of ticks across the rounds that ALL agents achieved convergence for Reinfocement Learning is around 6 times higher than Fictitious Play. Trade-off has to be made and 0.01 is used under the current setting.
4. We define effectiveness as below:

$$Effectiveness = 0.5 * \ convergence\ probability + \ 0.5 * convergence\ rate$$

$$where\ convergence\ rate = \left(1 - \frac{average\ number\ of\ ticks\ to\ converge}{30{,}000}\right)$$

* Here, we assume both convergence probability and convergence rate are equally important under the current setting.
* Effectiveness of Fictitious Play is $0.5 * 0.25 + 0.5 * \left(1 - \frac{1628}{30{,}000}\right) = 0.598$
* Effectiveness of Reinforcement Learning is $0.5 * 0.40 + 0.5 * \left(1 - \frac{9924}{30{,}000}\right) = 0.535$
* In conclusion, Fictitious Play is more effective based on our definition of "Effectiveness". Ultimately, this depends on the particular use case that we are interested in.
  - If achieving high convergence probability is crucial regardless of how long it takes, we can set higher weightage for convergence probability in the equation.

Problem 2

Part A : Model the problem as cooperative game (N, v) **(Specify what are *N* and *v*).**

1. ***N*** represents the set of players.
   * There are 4 agents in this game, i.e., N = { A, B, C, D }.
2. ***v*** represents the characteristic function.
   * v: 2^N → R
   * All possible coalition structures = {Ø, {A}, {B}, {C}, {D}, {A, B}, {A, C}, {A, D}, {B, C}, {B, D}, {C, D}, {A, B, C}, {A, B, D}, {A, C, D}, {B, C, D}, {A, B, C, D}}.
   * v(C) is the amount that the coalition C has to pay by working together and is defined as follows :

v(C) =	0	if C = {Ø}
	9	if C = {A}
	10	if C = {B}
	16	if C = {C}, {A, C}
	12	if C = {D}
	17	if C = {A, B}
	20	if C = {A, D}
	23	if C = {B, C}
	29	if C = {B, D}, {A, B, D}
	26	if C = {C, D}
	25	if C = {A, B, C}
	30	if C = {A, C, D}
	37	if C = {B, C, D}
	39	if C = {A, B, C, D}

Part B : Calculate the Shapley value for all agents in the "grand coalition" : $\Phi_i(N, v), i \in \{A, B, C, D\}$.

1. Shapley value for agent A.

   * Computation steps:

A X X X	X A X X	X X A X	X X X A
[v({A}) – v({Ø})] x 3! = 54	[v({A, B}) – v({B})] x 2! = 14 [v({A, C}) – v({C})] x 2! = 0 [v({A, D}) – v({D})] x 2! = 16	[v({A, B, C}) – v({B, C})] x 2! = 4 [v({A, B, D}) – v({B, D})] x 2! = 0 [v({A, C, D}) – v({C, D})] x 2! = 8	[v({A, B, C, D}) – v({B, C, D})] x 3! = 12

   * (54 + 14 + 0 + 16 + 4 + 0 + 8 + 12) / 4! = **4.5**

2. Shapley value for agent B.

   * Computation steps:

B X X X	X B X X	X X B X	X X X B
[v({B}) – v({Ø})] x 3! = 60	[v({B, A}) – v({A})] x 2! = 16 [v({B, C}) – v({C})] x 2! = 14 [v({B, D}) – v({D})] x 2! = 34	[v({B, A, C}) – v({A, C})] x 2! = 18 [v({B, A, D}) – v({A, D})] x 2! = 18 [v({B, C, D}) – v({C, D})] x 2! = 22	[v({A, B, C, D}) – v({A, C, D})] x 3! = 54

* (60 + 16 + 14 + 34 + 18 + 18 + 22 + 54) / 4! = **9.83**

3. Shapley value for agent C.

* Computation steps:

C X X X	X C X X	X X C X	X X X C
[v({C}) – v({Ø})] x 3! = 96	[v({C, A}) – v({A})] x 2! = 14 [v({C, B}) – v({B})] x 2! = 26 [v({C, D}) – v({D})] x 2! = 28	[v({C, A, B}) – v({A, B})] x 2! = 16 [v({C, A, D}) – v({A, D})] x 2! = 20 [v({C, B, D}) – v({B, D})] x 2! = 16	[v({A, B, C, D}) – v({A, B, D})] x 3! = 60

* (96 + 14 + 26 + 28 + 16 + 20 + 16 + 60) / 4! = **11.5**

4. Shapley value for agent D.

* Computation steps:

D X X X	X D X X	X X D X	X X X D
[v({D}) – v({Ø})] x 3! = 72	[v({D, A}) – v({A})] x 2! = 22 [v({D, B}) – v({B})] x 2! = 38 [v({D, C}) – v({C})] x 2! = 20	[v({D, A, B}) – v({A, B})] x 2! = 24 [v({D, A, C}) – v({A, C})] x 2! = 28 [v({D, B, C}) – v({B, C})] x 2! = 28	[v({A, B, C, D}) – v({A, B, C})] x 3! = 84

* (72 + 22 + 38 + 20 + 24 + 28 + 28 + 84) / 4! = **13.17**

**** Part C : Find the socially optimal coalition structure CS* (show the steps as in the slides).***

* Computation steps (Dynamic Programming):

Step	Coalition	Evaluations Performed Before Setting f		t	f
1	{A} {B} {C} {D}	v({A}) = 9 v({B}) = 10 v({C}) = 16 v({D}) = 12		{A} {B} {C} {D}	9 10 16 12
2	{A, B} {A, C} {A, D} {B, C} {B, D} {C, D}	v({A, B}) = 17 v({A, C}) = 16 v({A, D}) = 20 v({B, C}) = 23 v({B, D}) = 29 v({C, D}) = 26	f({A}) + f({B}) = 19 f({A}) + f({C}) = 25 f({A}) + f({D}) = 21 f({B}) + f({C}) = 26 f({B}) + f({D}) = 22 f({C}) + f({D}) = 28	{A, B} {A, C} {A, D} {B, C} {B} {D} {C, D}	17 16 20 23 22 26
3	{A, B, C} {A, B, D} {A, C, D} {B, C, D}	v({A, B, C}) = 25 v({A, B, D}) = 29 v({A, C, D}) = 30 v({B, C, D}) = 37	f({A}) + f({B, C}) = 32 f({B}) + f({A, C}) = 26 f({C}) + f({A, B}) = 33 f({A}) + f({B, D}) = 31 f({B}) + f({A, D}) = 30 f({D}) + f({A, B}) = 29 f({A}) + f({C, D}) = 35 f({C}) + f({A, D}) = 36 f({D}) + f({A, C}) = 28 f({C}) + f({B, D}) = 38 f({B}) + f({C, D}) = 36 f({D}) + f({C, B}) = 35	{A, B, C} {A, B, D} {D} {A, C} {D} {C, B}	25 29 28 35
4	{A, B, C, D}	v({A, B, C, D}) = 43	f({A}) + f({B, C, D}) = 44 f({B}) + f({A, C, D}) = 38 f({C}) + f({B, A, D}) = 45 f({D}) + f({B, C, A}) = 37	f({A, B}) + f({C, D}) = 43 f({A, C}) + f({B, D}) = 38 f({B, C}) + f({A, D}) = 43	{D} {A, B, C}	37

* Based on the Dynamic Programming algorithm, the socially optimal coalition structure is **{{D}, {A, B, C}}**, the total cost is **37**.

* ***Part D : For the Shapley value $\Phi_i(N, v)$ calculated in Part B, is it in the core of the (N, v)? Prove your answer using the definition of core.***
  * The definition of the core of a game is the set of all stable outcomes, i.e., outcomes that no coalition wants to deviate from.
  * However, for this particular game, we define the core as follows (instead of following the slides) as the agents aim to minimize the cost:

$$\text{core}(G) = \{(CS, \underline{x}) | \sum_{i \in C} x_i \leq v(C) \text{ for any } C \subseteq N\}$$

  * The grand coalition with outcome of (({A, B, C, D}), (4.5, 9.83, 11.5, 13.17)) is **NOT** in the core of (N, v) as there exists a coalition C, such that the $\sum_{i \in C} x_i > v(C)$.
  * The counter example would be agents A, B, C can form a coalition, i.e., {A, B, C} and v({A, B, C}) = 25.
  * Based on the payoff vector of the grand coalition (for agents A, B, C), gives 4.5 + 9.83 + 11.5 = **25.83**.

End of Assignment 3